一、代码如下：

#include <stdio.h>  
#include <stdlib.h>  
typedef struct  
{  
float \*base;  
float \*top;  
int StackSize;  
}Stack;  
  
   
void Createstack(Stack &s)  
{  
s.base=(float \*)malloc(100 \* sizeof(float));  
s.top=s.base;  
s.StackSize=100;  
  
}  
  
float top(Stack s)  
{  
if(s.top==s.base)  
{  
return -1;  
printf("error!\n");                
}  
return \*(s.top-1);  
}  
  
void push(Stack &s,float elem)  
{  
if(s.top-s.base>=s.StackSize)  
{  
  
s.base=(float \*)realloc(s.base,(s.StackSize+10)\*sizeof(float));  
s.top=s.base+s.StackSize;  
    s.StackSize+=10;  
}  
 \*s.top++ = elem;  
}  
  
  
void pop(Stack &s)  
{  
if(s.top==s.base)  
{  
printf("error!\n");  
return ;  
}  
s.top--;  
}  
  
  
int operate(char c)  
{  
switch(c)  
{  
case '+' :return 1;  
case '-' :return 1;  
case '\*' :return 1;  
case '/' :return 1;  
default :return 0;  
}  
}  
  
float cal(float a,float b,char c)  
{  
switch(c)  
{  
case '+' :return a+b;  
case '-' :return a-b;  
case '\*' :return a\*b;  
case '/' :return a/b;  
}  
}  
  
   
int main()  
{  
float a,b,result;  
Stack s;  
Createstack(s);  
char c1='0';  
    printf("输入后缀表达式：");   
while(c1!='A')  
{  
c1=getchar();  
if(!operate(c1))  
{  
  
push(s,(float) (c1-48));  
}  
else  
{  
b=top(s);  
pop(s);  
a=top(s);  
pop(s);  
result=cal(a,b,c1);  
push(s,result);  
}  
  
}  
printf("%f\n",result);  
  
return 0;  
}

1. 实验总结：

此次实验遇到的问题比较多，主要是对于先序遍历、中序遍历、后续遍历不够了解。三者的遍历方式比较容易混淆，在对课本进行查阅后熟悉了三种遍历的用法和区别，现已能够很熟练的使用三种遍历去遍历二叉树。对于课堂上的重点问题需要及时复习巩固，以免遗忘。